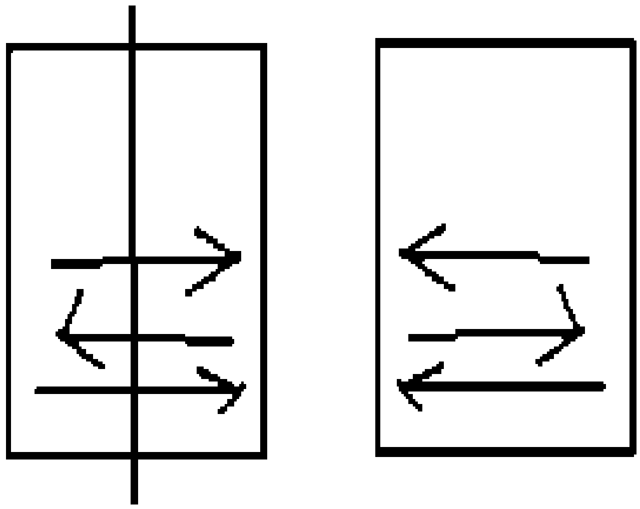
1 Mowing Yards (part 2)

1.1 Starting at Different Corners: Going in Different Directions

Up until now all of our mowing starts in the lower left corner of a square or rectangular lawn. Maybe we would like to start in the upper left and move in either direction, or even in the upper right. One way to do this would be to re-do all we have done before, only making everything start at some other point. That would involve repeating a considerable amount of effort when we know intuitively that these problems and their solutions are all essentially equivalent. "Lower left corner" obviously has no intrinsic meaning to a square. By rotating it any point could be "lower" and by turning it over or looking at its reflection in a mirror the notions of "left" and "right" are reversed. By geometrically (visually) applying these "symmetries" of the square we realize that the analysis of starting at any particular corner and heading in either possible direction can be described in terms of what happens in the case we have already done. What we need to do is understand how to quantify the symmetries of the problem mathematically and apply the corresponding mathematical symmetries to the work we have already done.

For instance the diagram below shows what we are currently doing: starting at the lower left abd proceeding to the right. We could start at the lower right and proceed to the left simply by reflecting everything through a line perpendicular to the x-axis which bisects the yard. Suppose that we have arranged things so that the bisecting lineis in fact the y-axis



Then the analytic (algebraic) realization of this reflection is simply reversing the sign of the x-coordinates. That is if P is the point [x,y] then the point which is the reflection of P through the y-axis is simply [-x,y]. What this means is that we can convert the algebra of mowing from left to right to that of mowing from right to left simply by changing some signs. This promises to be far more simple that re-doing all of our previous work.

Of course in general the y-axis isn't a line of symmetry of a problem but our powerful idea of symmetry tells us how to handle this: if there is a line of symmetry then since we can intuitively see that that all lines in the plane are "equivalent" we can take the alebra which expersses the equivalence of the given line and the y-axis and use it to convert the analysis of the simple case into that of the one which interests us. For example, if the line of symmetry isn't the y-axis but is parallel to it we can simply apply a shift which moves the line of symmetry to the y-axis, then refelect through the y-axis, then shift back. If the shift which moves our line to the y-axis is [SHIFTX, SHIFTY] then the recipe would simply be:

[X, Y] -> [X + SHIFTX, Y + SHIFTY] -> [-X - SHIFTX, Y + SHIFTY] -> [-X -2* SHIFTX, Y]

Applying this transformations to the vertices of a polygon (lawn) would then reflect that polygon through the given line. For instance if Larry's lawn is

Larryslawn:=[[0,0],[3,0],[3,1], [0,1]];

and we want to refelect Larry's lawn through its centerline, x=1.5 we would use a shift of [-1.5,0]. Then

```
SHIFTX:=-1.5; SHIFTY:=0;
reflectedLarryslawn:=[[-0-2*(SHIFTX),0],[-3-2*SHIFTX,0],[-3-2*SHIFTX,
1], [-0-2*SHIFTX,1]];
```

Now if we view both Larryslawn and reflected Larryslawn, one red and one blue we see:

```
LL:=plots[polygonplot](Larryslawn,
color=blue);
RLL:=plots[polygonplot](reflectedLarryslawn, color=red);
plots[display]([LL,RLL],scaling=constrained);
```

We are missing red one. Actually not, lets reverse the order

```
plots[display]([
RLL,LL],scaling=constrained);
```

The point is that the reflection transformed Larry's lawn **onto itself**, reversing left and right. The red and blue lawns occupy the same space.

Now we can use this idea to "reflect" our previous mowing. The primary idea, which we have actually used before, is that our lawnmower code employs the coordinates **only through the** `lawn function. Thus all we will` need to do is modify lawn to accomplish reflection. Lets first write a version of lawn that reflects everything through the line x=3 to check that we know how to do it.

```
lawnreflxeq3:= proc (v1,v2,v3,v4, clr )
local w1,w2,w3,w4,shift;
shift:=[-3,0]:
w1:=[-v1[1]-2*shift[1],v1[2]]:
w2:=[-v2[1]-2*shift[1],v2[2]]:
w3:=[-v3[1]-2*shift[1],v3[2]]:
w4:=[-v4[1]-2*shift[1],v4[2]]:
plots[polygonplot]([w1,w2,w3,w4],color=clr,style=patch,scaling=constra
ined ):
end:

lawnreflxeq3([3,0],[7,0],[7,2],[3,2],
```

```
red);
```

We modify lawnmower3 simply by changing lawnmowerwithshift to lawnreflxeq3.

```
lawnmower4:=proc(Lngth, Wdth, MOWINGTIME,
MOVIELENGTH, NUMSTRIPS, SHIFT )
local movie, YARD, MOWEDYARD, MRWDTH, TIME, WHICHSTRIP, FRACSTRIP,
MOWN, FRAME, LASTFRAME ;
movie:=NULL:
YARD:=lawnreflxeq3([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
aquamarine,SHIFT):
MOWEDYARD:=lawnreflxeq3([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
green,SHIFT):
MRWDTH:=Wdth/NUMSTRIPS:
for TIME from 0 to MOVIELENGTH-1 do
if TIME <=MOWINGTIME then
WHICHSTRIP:= floor(NUMSTRIPS*TIME/MOWINGTIME);
FRACSTRIP:=TIME*(NUMSTRIPS/MOWINGTIME) - WHICHSTRIP;
MOWN:=lawnreflxeq3(
[0,0],[Lngth,0],[Lngth,MRWDTH*WHICHSTRIP],[0,MRWDTH*WHICHSTRIP]
,green,SHIFT):
if WHICHSTRIP mod 2 = 0 then
FRAME:=plots[display]([MOWN, lawnreflxeq3( [0,MRWDTH*(WHICHSTRIP )]
, [Lngth *FRACSTRIP,MRWDTH*(WHICHSTRIP )
],[Lngth*FRACSTRIP,MRWDTH*(WHICHSTRIP+1) ],[0,MRWDTH*(WHICHSTRIP+1)],
green,SHIFT ) ],YARD):
else
FRAME:=plots[display]([MOWN, lawnreflxeq3([Lngth *(
1-FRACSTRIP),MRWDTH*(WHICHSTRIP )], [Lngth,MRWDTH*(WHICHSTRIP )],
[Lngth,MRWDTH*(WHICHSTRIP+1)], [Lngth *( 1-
FRACSTRIP),MRWDTH*(WHICHSTRIP+1) ] , green,SHIFT ) ],YARD):
fi:
movie:=movie,FRAME:
else
movie:=movie,MOWEDYARD:
fi:
od:
if MOVIELENGTH >= MOWINGTIME then LASTFRAME:=MOWEDYARD:
movie:=movie,LASTFRAME:plots[display]([movie],insequence=true):
fi:
[movie];
end;

MOVIE:=lawnmower4(30,20,25,20,10,[0,0] ):

plots[display]( MOVIE,insequence=true);
```

Thus we see that by altering the "lawn" component of our lawnmower program we can produce lawnmower starting at other corners and moving in other directions. Now to make things a little simpler we will concentrate on a single square, 10 by 10 lawn which is symmetric about the origin and return to the original problem. We want to make a movie of Sam and Bill mowing the **same** lawn.

1.2 Sam and Bill Mowing the Same Lawn

Our lawn will be:

```
SQUAREYARD:=[[-5,-5],[5,-5],[5,5],[-5,5]]:
```

We make versions of lawn which will reflects through the x-axis or the y-axis

```
SQUAREYARD:=[[-5,-5],[5,-5],[5,5],[-5,5]]:
lawnx:= proc (v1,v2,v3,v4, clr,shift ) local w1,w2,w3,w4 ;
w1:=v1+shift:
w2:=v2+shift:
w3:=v3+shift:
w4:=v4+shift:
w1:=[-w1[1] ,w1[2]]:
w2:=[-w2[1],w2[2]]:
w3:=[-w3[1] ,w3[2]]:
w4:=[-w4[1] ,w4[2]]:
plots[polygonplot]([w1,w2,w3,w4],color=clr,style=patch,
scaling=constrained ):
end:

lawny:= proc (v1,v2,v3,v4, clr,shift ) local
w1,w2,w3,w4 ;
w1:=v1+shift:
w2:=v2+shift:
w3:=v3+shift:
w4:=v4+shift:
w1:=[ w1[1] ,-w1[2]]:
w2:=[ w2[1] ,-w2[2]]:
w3:=[ w3[1] ,-w3[2]]:
w4:=[ w4[1] ,-w4[2]]:
plots[polygonplot]([w1,w2,w3,w4],color=clr,style=patch,
scaling=constrained ):
end:
```

Now we can make mower who starts in different corners and proceeds in different directions simply by changing the version of "lawn" as we did in lawnmower4. Lets make a mower who starts at the lower left and mows SQUAREYARD. All we do is grab a copy of lawnmower4 and change the "lawn" function to "lawnx"

```
lawnmower4x:=proc(Lngth, Wdth, MOWINGTIME,
MOVIELENGTH, NUMSTRIPS, SHIFT )
local movie, YARD, MOWEDYARD, MRWDTH, TIME, WHICHSTRIP, FRACSTRIP,
MOWN, FRAME, LASTFRAME ;
movie:=NULL:
YARD:=lawnx([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
aquamarine,SHIFT):
MOWEDYARD:=lawnx([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
green,SHIFT):
```

```
MRWDTH:=Wdth/NUMSTRIPS:
for TIME from 0 to MOVIELENGTH-1 do
if TIME <=MOWINGTIME then
WHICHSTRIP:= floor(NUMSTRIPS*TIME/MOWINGTIME);
FRACSTRIP:=TIME*(NUMSTRIPS/MOWINGTIME) - WHICHSTRIP;
MOWN:=lawnx(
[0,0],[Lngth,0],[Lngth,MRWDTH*WHICHSTRIP],[0,MRWDTH*WHICHSTRIP]
,green,SHIFT):
if WHICHSTRIP mod 2 = 0 then
FRAME:=plots[display]([MOWN, lawnx( [0,MRWDTH*(WHICHSTRIP )] ,
[Lngth *FRACSTRIP,MRWDTH*(WHICHSTRIP )
],[Lngth*FRACSTRIP,MRWDTH*(WHICHSTRIP+1) ],[0,MRWDTH*(WHICHSTRIP+1)],
green,SHIFT ) ],YARD):
else
FRAME:=plots[display]([MOWN, lawnx([Lngth *(
1-FRACSTRIP),MRWDTH*(WHICHSTRIP )], [Lngth,MRWDTH*(WHICHSTRIP )],
[Lngth,MRWDTH*(WHICHSTRIP+1)], [Lngth *( 1-
FRACSTRIP),MRWDTH*(WHICHSTRIP+1) ] , green,SHIFT ) ],YARD):
fi:
movie:=movie,FRAME:
else
movie:=movie,MOWEDYARD:
fi:
od:
if MOVIELENGTH >= MOWINGTIME then LASTFRAME:=MOWEDYARD:
movie:=movie,LASTFRAME:plots[display]([movie],insequence=true):fi:
[movie];
end;
```

Now **lawn mower lower right,** `LMLR` is

```
LMLR:=proc(MOWINGTIME, MOVIELENGTH,
NUMSTRIPS)
lawnmower4x (10, 10, MOWINGTIME, MOVIELENGTH, NUMSTRIPS, [-5,-5]
)end;
joe:=LMLR(20,20,9):

plots[display](joe,insequence=true);
```

A left upper mower, `LMUL would use` lawny . Just make a lawnmower4y

```
lawnmower4y:=proc(Lngth, Wdth, MOWINGTIME,
MOVIELENGTH, NUMSTRIPS, SHIFT )
local movie, YARD, MOWEDYARD, MRWDTH, TIME, WHICHSTRIP, FRACSTRIP,
MOWN, FRAME, LASTFRAME ;
movie:=NULL:
YARD:=lawny([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
aquamarine,SHIFT):
MOWEDYARD:=lawny([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
green,SHIFT):
MRWDTH:=Wdth/NUMSTRIPS:
for TIME from 0 to MOVIELENGTH-1 do
```

```
if TIME <=MOWINGTIME then
WHICHSTRIP:= floor(NUMSTRIPS*TIME/MOWINGTIME);
FRACSTRIP:=TIME*(NUMSTRIPS/MOWINGTIME) - WHICHSTRIP;
MOWN:=lawny(
[0,0],[Lngth,0],[Lngth,MRWDTH*WHICHSTRIP],[0,MRWDTH*WHICHSTRIP]
,green,SHIFT):
if WHICHSTRIP mod 2 = 0 then
FRAME:=plots[display]([MOWN, lawny( [0,MRWDTH*(WHICHSTRIP )] ,
[Lngth *FRACSTRIP,MRWDTH*(WHICHSTRIP )
],[Lngth*FRACSTRIP,MRWDTH*(WHICHSTRIP+1) ],[0,MRWDTH*(WHICHSTRIP+1)],
green,SHIFT ) ],YARD):
else
FRAME:=plots[display]([MOWN, lawny([Lngth *(
1-FRACSTRIP),MRWDTH*(WHICHSTRIP )], [Lngth,MRWDTH*(WHICHSTRIP )],
[Lngth,MRWDTH*(WHICHSTRIP+1)], [Lngth *( 1-
FRACSTRIP),MRWDTH*(WHICHSTRIP+1) ] , green,SHIFT ) ],YARD):
fi:
movie:=movie,FRAME:
else
movie:=movie,MOWEDYARD:
fi:
od:
if MOVIELENGTH >= MOWINGTIME then LASTFRAME:=MOWEDYARD:
movie:=movie,LASTFRAME:plots[display]([movie],insequence=true):
fi:
[movie];
end;
```

Now use it to define **lawn mower upper left**, `LMUL .  Note the similarity to` LMLR

```
LMUL:=proc(MOWINGTIME, MOVIELENGTH,
NUMSTRIPS)
lawnmower4y (10, 10, MOWINGTIME, MOVIELENGTH, NUMSTRIPS, [-5,-5]
)end;
```

```
joe:=LMUL(20,20,9)
```

```
plots[display](joe,insequence=true);
```

Our original `lawnwithshift` gives us **lawn mower lower left** LMLL but thats just lawnmower3

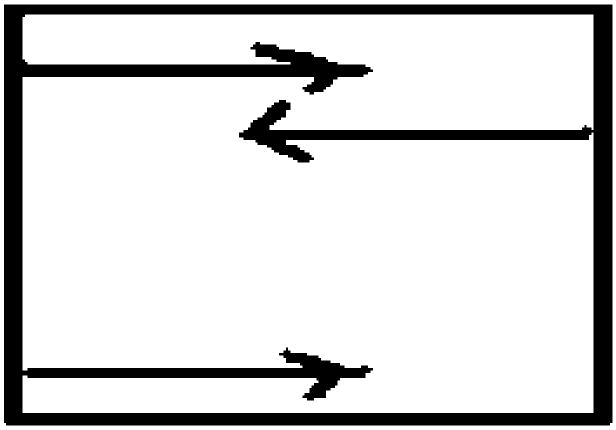
```
LMLL:=proc(MOWINGTIME, MOVIELENGTH,
NUMSTRIPS)
lawnmower3 (10, 10, MOWINGTIME, MOVIELENGTH, NUMSTRIPS, [-5,-5]
)end;
```

```
joe:=LMLL(20,20,9):
```

```
plots[display](joe,insequence=true);
```

Now we are getting close to having Sam and Joe cooperate on mowing the lawn. Knowing how long each takes to mow the lawn

separately we can calculate how long they should mow together. We can have one of them start at, say, the lower left and the other at the upper left and have them each mow for the computed time.



For instance suppose Sam mows the lawn in 30 minutes and Bill mows it in 40. Then if together they mow in TOGETHERTIME we have the equation eq:

```
eq:=1/TOGETHERTIME = 1/30 + 1/40;
ans:=solve(eq,TOGETHERTIME);
ans:=evalf(ans)
```

This says that if we, for instance make a 17 frame movie of Bill mowing from the top and Sam mowing from the bottom then we should be able to put them together and make a movie of both mowing most of the lawn together together. Lets make the two separate ones first.

```
BILLMOWING:=LMLL(40,17,7):

SAMMOWING:=LMUL(30,17,7):

plots[display](BILLMOWING,insequence=true);

plots[display](SAMMOWING,insequence=true);
```

Recall that we know how to run two movies at simultaneously. This should also allow us to overlay one upon another. (Note this isn't going to work exactly).

```
movie:=seq(plots[display]([SAMMOWING[i],BILLM
OWING[i]]),
i=1..nops(SAMMOWING))

plots[display]([movie],insequence=true)
```

There are two problems. One is that the movie only has our mowers mowing 17 time units while it takes a little more than that to finish the job. The other is that we have one movie playing on top of the other SAMMOWING is on top of BILLMOWING. This would be ok except that each frame of SAMMOWING also puts down the unnown lawn as a base. Thus we need a slightly modified version of SAMMOWING in which the base isn't put down. We just need to go back and modify lawnmower4y to lawnmower4y without background. The reader should compare the following to lawnmower4y. All we have done is delete the background (YARD) from each frame to get "lawnmower4 without background". Note that the deletions are accomplished by "commenting" out the definition of "YARD" and references to "YARD" in "lawnmower4y".

```
lm4ywobg:=proc(Lngth, Wdth, MOWINGTIME,
MOVIELENGTH, NUMSTRIPS, SHIFT )
local movie, YARD, MOWEDYARD, MRWDTH, TIME, WHICHSTRIP, FRACSTRIP,
MOWN, FRAME, LASTFRAME ;
movie:=NULL:
#YARD:=lawnreflxeq3([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
aquamarine,SHIFT):
MOWEDYARD:=lawny([0,0],[Lngth , 0],[Lngth ,Wdth ],[ 0,Wdth ],
green,SHIFT):
MRWDTH:=Wdth/NUMSTRIPS:
for TIME from 0 to MOVIELENGTH-1 do
if TIME <=MOWINGTIME then
WHICHSTRIP:= floor(NUMSTRIPS*TIME/MOWINGTIME);
FRACSTRIP:=TIME*(NUMSTRIPS/MOWINGTIME) - WHICHSTRIP;
MOWN:=lawny(
[0,0],[Lngth,0],[Lngth,MRWDTH*WHICHSTRIP],[0,MRWDTH*WHICHSTRIP]
,green,SHIFT):
if WHICHSTRIP mod 2 = 0 then
FRAME:=plots[display]([MOWN, lawny( [0,MRWDTH*(WHICHSTRIP )] ,
[Lngth *FRACSTRIP,MRWDTH*(WHICHSTRIP )
],[Lngth*FRACSTRIP,MRWDTH*(WHICHSTRIP+1) ],[0,MRWDTH*(WHICHSTRIP+1)],
green,SHIFT ) ] # ,YARD
):
else
FRAME:=plots[display]([MOWN, lawny([Lngth *(
1-FRACSTRIP),MRWDTH*(WHICHSTRIP )], [Lngth,MRWDTH*(WHICHSTRIP )],
[Lngth,MRWDTH*(WHICHSTRIP+1)], [Lngth *( 1-
FRACSTRIP),MRWDTH*(WHICHSTRIP+1) ] , green,SHIFT ) ] #,YARD
):
fi:
movie:=movie,FRAME:
else
movie:=movie,MOWEDYARD:
fi:
od:
if MOVIELENGTH >= MOWINGTIME then LASTFRAME:=MOWEDYARD:
movie:=movie,LASTFRAME:plots[display]([movie],insequence=true):
fi:
[movie];
end;
```

Now we can make the associated "lawn mower upper left without background"

```
LMULwobg:= proc(MOWINGTIME, MOVIELENGTH,
NUMSTRIPS)
lm4ywobg (10, 10, MOWINGTIME, MOVIELENGTH, NUMSTRIPS, [-5,-5]
)end;

SAMMOWINGwobg:=LMULwobg(30,17,7):

movie:=seq(plots[display]([SAMMOWINGwobg[i],
BILLMOWING[i]]),i=1..nops(SAMMOWING)):
plots[display]([movie],insequence=true);
```

Now the obvious way to finish is to have the last frame be the completely mown yard. Thats of course easy to lay in. We called it SQUAREYARD.

```
MOWNSQUAREYARD:=plots[polygonplot](SQUAREYARD
,
color=green, scaling=constrained):
```

Now using the previously defined "movie"

```
movie:=movie,MOWNSQUAREYARD:
plots[display]([movie],insequence=true);
```

This isn't completely satisfactory, however because we don't get to see the two lawnmowers come together at the point where each has exactly done his part. There are other problems, too. Depending on the number of strips it might be necessary for Sam and Bill to

start at diagonally opposite corners. For instance, suppose our current problem used 3 strips rather than 7.

```
BILLMOWING:=LMLL(40,17,3):
SAMMOWINGwobg:=LMULwobg(30,17,3):
movie:=seq(plots[display]([SAMMOWINGwobg[i],
BILLMOWING[i]]),i=1..nops(SAMMOWINGwobg)):
plots[display]([movie],insequence=true);
```

What has happened here is that the faster mower got to the middle strip and started mowing it ahead of the slower fellow. In order to keep this from happening they should have started at antipotal corners. In other word we need a version of "lawn" which starts at the upper right corner. We need an "LMURwobg" word to handle this situation. Thats easy enough, if we reflect through both the x and y axes with a "lawnxy" we can easily make one.

1.2.1 Exercise: *Make a "lawnxy" word and use it to create a mower who starts in the upper right corner and mows toward the y-axis. Then use this to make a movie of Sam and Bill mowing a three-strip lawn in which they don't try to mow the same grass.*

1.2.2 Mathematics Problem: *Sam and Bill have a square lawn which is 100 by 100 meters. They have two lawn mowers each of which cuts a path 1/2 meter wide. Using one of the mowers Sam can mow the lawn in 2 hours and Bill can mow it in 3. Using the algebra they are studying in school they conclude that using both mowers they should be able to mow the lawn in an hour and 12 minutes.*

Where does each start and how does he proceed so that neither ever passes his mower over grass that the other has already mown?

1.2.3 Problem Variation:*(Emptying The Swimming Poo***l)** *The "mowing problem" is really about adding rates of things that happen. A simpler problem to visualize might be that of emptying a swimming pool. Suppose there are two drains. One can empty the pool in 24 hours, the other in 36. Make a two dimensional animation of the pool emptying through each drain and the two drains together. Make a movie which shows the three cases side by side.*

1.2.4 *Start: For the moment assume the pool is constant depth. There isn't much difference between a lawn and a (vertical section of a pool). The "empty" pool could be the "unmown" yard and the "mown" grass the water in the pool (or visa versa). Changing colors to grey and blue would promote the image. The draining pool would not involve any partial strips so most of the difficulty of the mowing problem would not be present.*

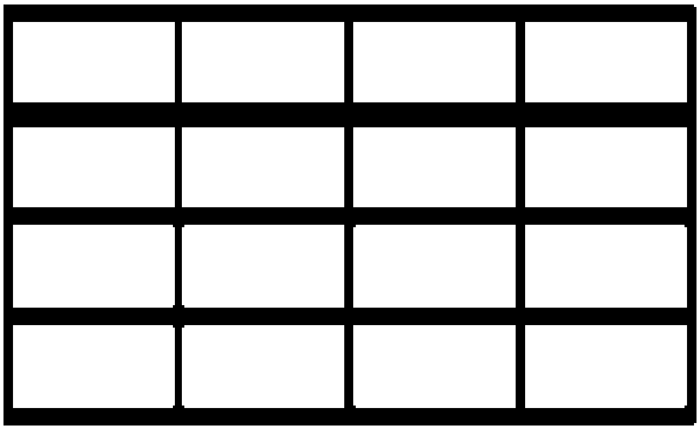
1.3 EXERCISES

The following exercises emphasize the idea of (when possible) isolating the coordinate-dependent components in a small number of Maple words and then modifying only those to take advantage of symmetry in a problem. In this case we will use a simplified version of CPB (chromatic brick placer) which we previously developed. This is a two dimensional version which places a horzontal brick of a specified color with its lower left corner at a specified point.

```
CBP2d:= proc ( pt ,clr )
local v1, v2, v3, v4, brick;
v1 := [pt[1],pt[2]];
v2 := [pt[1]+8, pt[2] ]; v3 := [pt[1]+8, pt[2]+4 ];
v4 := [pt[1], pt[2] +4]; brick := [ v1, v2, v3, v4];
plots[polygonplot](brick,color=clr,style=patch,
thickness=3,scaling=constrained ) end;

CBP2d([0,3],red);
```

Now we develop a word which lays down a small brick walls (say four bricks wide by five bricks high) which have the following form:



We make a "small brick wall" procedure called SBW

```
SBW:=proc(wide,high) local movie,frame,i,j;
movie:=NULL; frame:=NULL;
for i from 0 to wide-1 do
for j from 0 to high-1 do
frame:=plots[display](frame,CBP2d([8*i,4*j],red)):
movie:=movie,frame:
od:
od:
plots[display]([movie],insequence=true);
end;
```

```
SBW(4,5);
```

Note that the only part of `SBW` which uses the coordinates is PBC2d. Thus by modifiying the way PBC2d uses coordinates we can change the manner in which the bricks are set down.

1.3.1 Exercise: *Modify CPB2d so that SBW lays its bricks starting at the left.*

HINT: The wall is symmetric about the line x=(8*wide)/2 (at least the set of lower left vertices is). As noted at the beginning of the chapter the algebraic transformation which reflects the plane about the line x=wide/2 is

`[ X, Y ] -> [ X -8*wide/2, Y ]` -> [-X +8*wide/2, Y] -> [-X +2* 8*wide/2, Y]

Thus if PBC2d originally places a brick at [x,y] we now want it to place it at [-x+8*wide,y].

This means that if we change only one line in SBW we can produce a newSBW which does the exercise. One possibility for the changed line is:

```
frame:=plots[display](frame,CBP2d([-8*i+8*wide,4*j],red)):
```

```
newSBW:=proc(wide,high) local
movie,frame,i,j;
movie:=NULL; frame:=NULL;
for i from 0 to wide-1 do
for j from 0 to high-1 do
frame:=plots[display](frame,CBP2d([-8*i+8*wide,4*j],red)):
movie:=movie,frame:
od:
od:
plots[display]([movie],insequence=true);
end;

newSBW(5,4);
```

1.3.2 *EXERCISE:* `SBW(a,b)` *builds starts at the lower left, builds a stack "b" bricks high, then moves a step to the right and builds the next stack of "b" bricks, doing this "a" times. Modify SBW to produce SBWh (h for horizontal) which first lays a layer of "b" bricks, then starts over at the left and lays a second layer on top of this one, etc.*

HINT: This can be done by permuting two lines in SBW

1.3.3 *EXERCISE: Modify* `SBW to get SBWul ( "ul"` *for " upper left") so that SBWul(a,b) starts on the left and builds a stack of bricks of "b" bricks from the top down .*

HINT: This "stacking" (it might be a brick sidewalk) is the reflection of the action of SBL through the line y=4*b/2. The algebraic realization of this symmetry is [x,y]->[x,-y+2*(4*b/2)] = [x, -y+4*b].

1.3.4 *EXERCISE: Permute two lines in SBW to get SBWulh ("ulh" for " upper left, horizontal ") so that SBWul(a,b) starts on the upper left and builds "a" descending rows of "b" bricks.*

1.3.5 *EXERCISE: Modify* `SBW to get SBWbf` *(for "back and forth") so that SBWbf(a,b) starts laying bricks at the lower left, lays horizontal rows of "b" bricks, each time reversing direction to lay the next row.*

HINT: This switches between the (horizontal versions of) the actions of SBW and newSBW, depending on the parity of the row to be completed.

1.3.6 *EXERCISE: Make SBWs, a version of SBW which lays bricks in the familiar "shifted" pattern, one horizontal row at a time.*